Paweł Duraj

Informatyka stosowana - 2011/2012
Rok studiów: I

Obliczanie wartości wyrażenia podanego jako ciąg znaków.

1. Opis problemu:
   a) sprawdzenie poprawności zapisu wyrażenia podanego przez użytkownika;
   b) zamiana zapisu wyrażenia na odwrotną notację polską;
   c) obliczenie wartości wyrażenia;
2. Przyjęte założenia:
   a) podane przez użytkownika wyrażenie nie może składać się z większej liczby elementów niż wartość stałej zmiennej globalnej n (domyślnie 20), gdzie element to liczba lub operator lub znak nawiasu;
   b) wyrażenie nie może zawierać liczb: 0.001, 0.002, 0.003, 0.004, 0.005;
   c) podane wyrażenie musi być zapisane w postaci infiksowej;
   d) operatory brane pod uwagę to: +,-,*,/,^,();
3. Algorytmy rozwiązujące problem:

3.1 Sprawdzanie czy podane wyrażenie posiada tylko dozwolone znaki:

a) Specyfikacja wejścia/wyjścia:

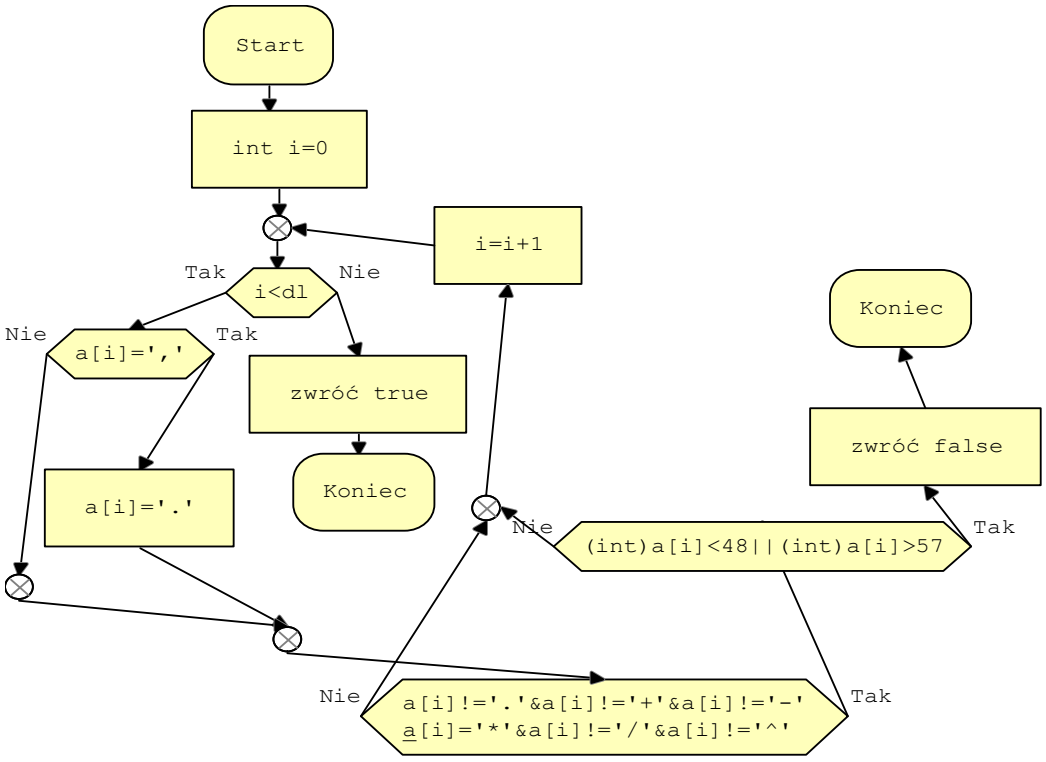
Dane: a – ciąg znaków typu string (przesłana do funkcji przez referencje)
dl – liczba znaków w zmiennej a

Wyjście: true – gdy znaki są poprawne
false – gdy zawiera niedozwolone znaki

b) Pseudokod:

Krok 1: dopóki i<dl (int i=0;i<dl;i++)
Krok 1.1: jeżeli a[i]==’i’
Krok 1.1.1: a[i]=’.’
Krok 1.2: jeżeli a[i]!=’.’&& a[i]!=’+’&& a[i]!=’-’&& a[i]!=’*’&& a[i]!=’/’&& a[i]!=’^’&& a[i]!=’(’&& a[i]!=’)’
Krok 1.2.1: jeżeli (int)a[i]<48||(int)a[i]>57
Krok 1.2.1.1: zwróć false

c) Schemat blokowy:



3.2 Sprawdzanie poprawności nawiasów:

a) Specyfikacja wejścia/wyjścia:

Dane: a – ciąg znaków typu string

dl – długość zmiennej a

Wyjście: true – gdy nawiasy są poprawne

false – gdy nawiasy nie są poprawnie zapisane

b) Pseudokod:

Krok 1: int k=0

Krok 2: dopóki i<dl (int i=0;i<dl;i++)

Krok 2.1: jeżeli a[i]==’(‘

Krok 2.1.1: k++

Krok 2.2: jeżeli a[i]==’)’

Krok 2.2.1: k --

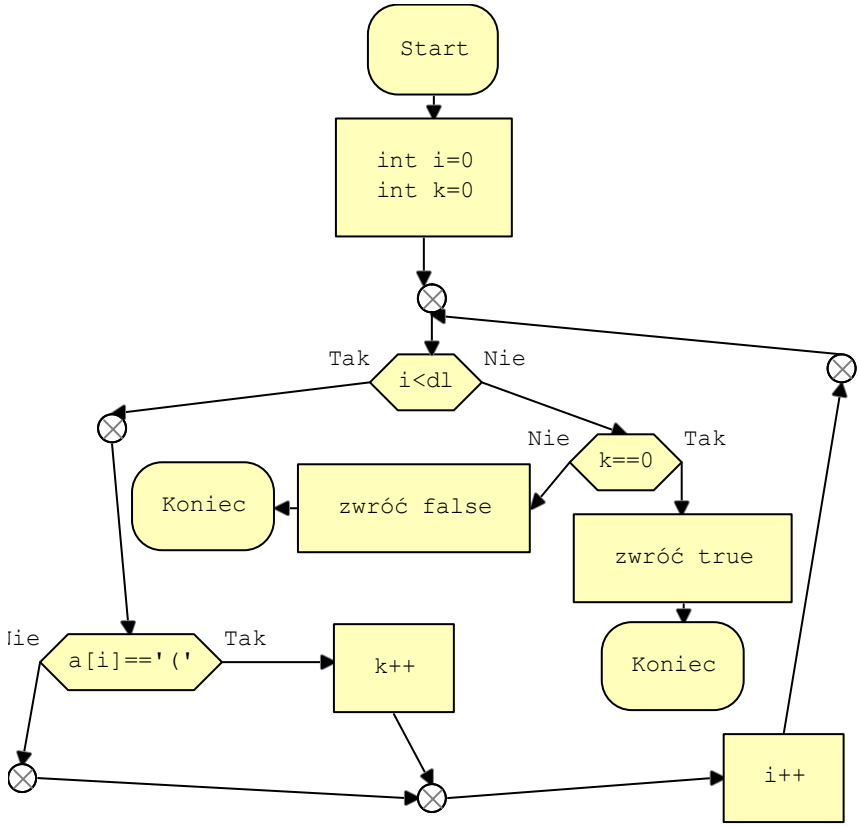
Krok 3: jeżeli k==0

Krok 3.1: zwróć true

Krok 4: w przeciwnym wypadku

Krok 4.1: zwróć false

c) Schemat blokowy:



3.3 Odpowiednie zapisanie liczb ujemnych:

a) Specyfikacja wejścia/wyjścia:

Dane: a – ciąg znaków typu string

dl – długość zmiennej a

Wyjście: b – zmienna string

b) Pseudokod:

Krok 1: int k=0,i=0,dlb=0

Krok 2: string b=””

Krok 3: jeżeli a[0]==’-‘

Krok 3.1: doklej ”(0-” do ciągu b

Krok 3.2: dopóki a[i]>47&&a[i]<58||a[i]==’.’ (i=1; a[i]>47&&a[i]<58||a[i]==’.’;i++)

Krok 3.2.1: doklej a[i] do b

Krok 3.3: doklej ”)” do ciągu b

Krok 4: dopóki i<dl (i;i<dl;i++)

Krok 4.1: jeżeli a[i]==’-‘&&a[i-1]==’(‘

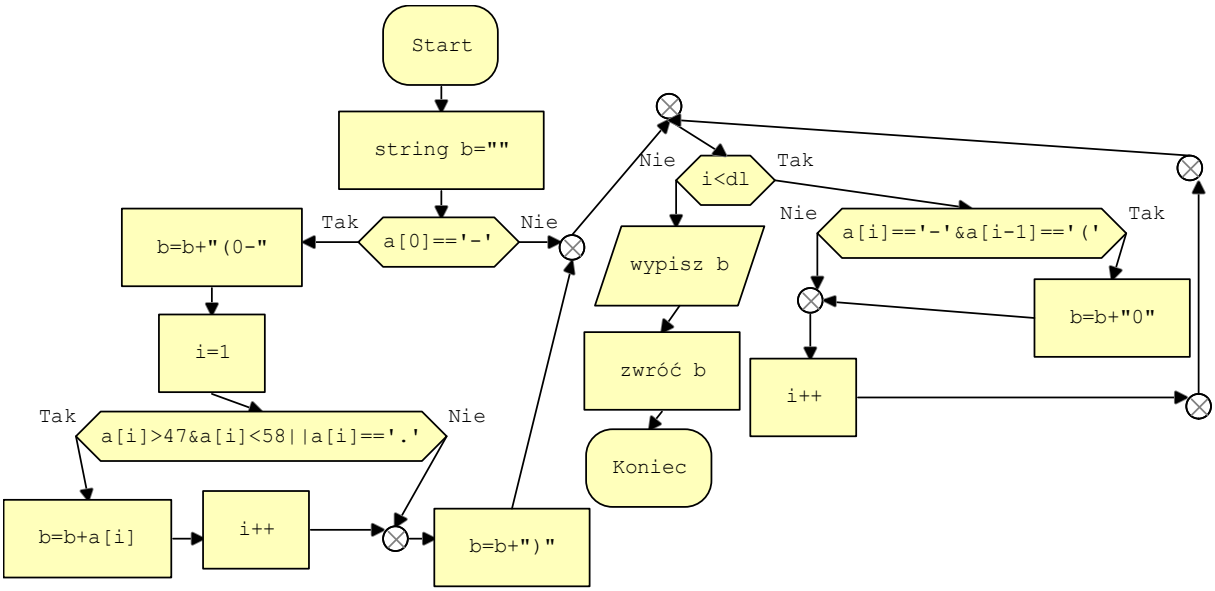
Krok 4.1.1: doklej ”0” do b

Krok 4.2: doklej a[i] do b

Krok 5: wypisz b

Krok 6: zwróć b

c) Schemat blokowy:



3.4 Wyznaczanie priorytetów działań:

a) Specyfikacja wejścia/wyjścia:
   Dane: znak - zmienna typu char
   Wyjście: liczba określająca priorytet
b) Pseudokod:
   Krok 1: jeżeli znak==’(‘
   Krok 1.1: zwróć 0
   Krok 2: jeżeli znak==’+‘
   Krok 2.1: zwróć 1
   Krok 3: jeżeli znak==’-‘
   Krok 3.1: zwróć 1
   Krok 4: jeżeli znak==’)‘
   Krok 4.1: zwróć 1
   Krok 5: jeżeli znak==’*‘
   Krok 5.1: zwróć 2
   Krok 6: jeżeli znak==’/‘
   Krok 6.1: zwróć 2
   Krok 7: jeżeli znak==’^‘
   Krok 7.1: zwróć 3

3.5 Rzutowanie operatorów:

a) Specyfikacja wejścia/wyjścia:
   Dane: a - znak typu char
         il - zmienna globalna typu int odpowiedzialna za zliczanie używanych indeksów tab
   Wyjście: x - liczba typu double
b) Pseudokod:
   Krok 1: double x
   Krok 2: jeżeli a==’+’
   Krok 2.1: x=0.001
   Krok 3: jeżeli a==’-’
   Krok 3.1: x=0.002
   Krok 4: jeżeli a==’*’
   Krok 4.1: x=0.003
   Krok 5: jeżeli a==’/’
   Krok 5.1: x=0.004
   Krok 6: jeżeli a==’^’
   Krok 6.1: x=0.005

3.6 Zamiana wyrażenia z postaci infiksowej na odwrotną notacje polską:

a) Specyfikacja wejścia/wyjścia

Dane: wyr – ciąg znaków typu string

dl – długość wyr

priorytet - funkcja ustalająca priorytet działania (algorytm 3.4)

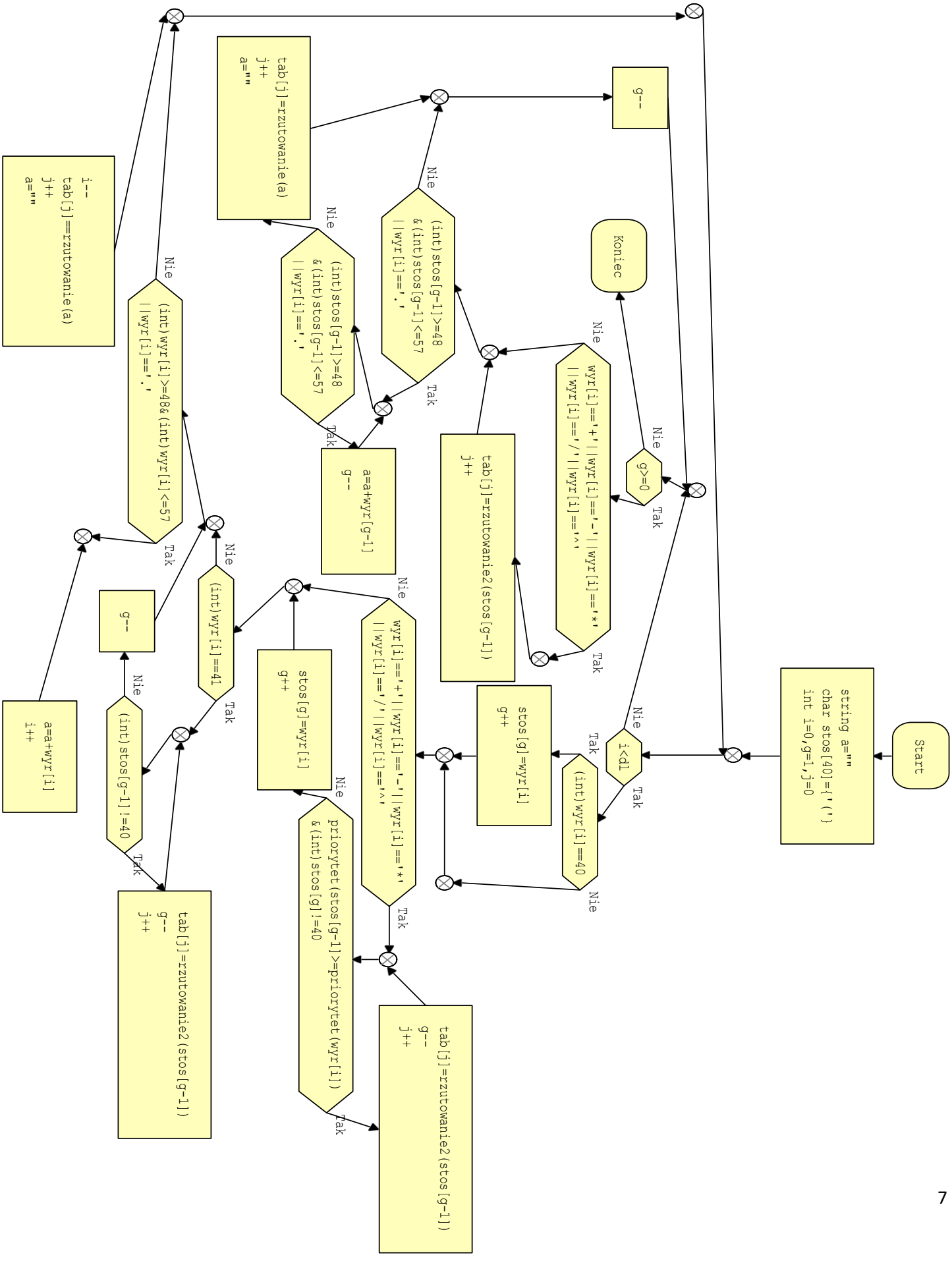
rzutowanie - funkcja zamieniająca ciąg znaków na liczbę

rzutowanie2 - funkcja zamieniająca znak operatora na liczbę (algorytm 3.5)

b) Pseudokod:

Krok 1: string a="", char stos[80]={'('}
Krok 2: int i,g=1,j=0
Krok 3: dopóki i<dl (i=0;i<dl;j++)
Krok 3.1: jeżeli (int)wyr[i]==40
Krok 3.1.1: stos[g]=wyr[i]
Krok 3.1.2: g++
Krok 3.2: jeżeli wyr[i]=='+'||wyr[i]=='-'||wyr[i]=='*'||wyr[i]=='/'||wyr[i]=='^'
Krok 3.2.1: dopóki priorytet(stos[g-1])>=priorytet(wyr[i])&&stos[g]!=40
Krok 3.2.1.1: tab[j]=rzutowanie2(stos[g-1])
Krok 3.2.1.2: g--
Krok 3.2.1.3: j++
Krok 3.2.2: stos[g]=wyr[i]
Krok 3.2.3: g++
Krok 3.3: jeżeli (int)wyr[i]==41
Krok 3.3.1: dopóki (int)stos[g-1]==40
Krok 3.3.1.1: tab[j]=rzutowanie2(stos[g-1])
Krok 3.3.1.2: g--
Krok 3.3.1.3: j++
Krok 3.3.2: g--
Krok 3.4: jeżeli (int)wyr[i]>=48&&(int)wyr[i]<=57||wyr[i]=='.'
Krok 3.4.1: dopóki (int)wyr[i]>=48&&(int)wyr[i]<=57||wyr[i]=='.'
Krok 3.4.1.1: doklej wyr[i] do a
Krok 3.4..1.2: i++
Krok 3.4.2: i--
Krok 3.4.3: tab[j]=rzutowanie(a)
Krok 3.4.4: j++
Krok 3.4.5: a=""
Krok 4: dopóki g>=0
Krok 4.1: jeżeli stos[g-1]=='+'|| stos[g-1]=='-'|| stos[g-1]=='*'|| stos[g-1]=='/'|| stos[g-1]=='^'
Krok 4.1.1: tab[j]=rzutowanie2(stos[g-1])
Krok 4.1.2: j++
Krok 4.2: jeżeli (int)stos[g-1]>=48&&(int)stos[g-1]<=57||wyr[i]=='.'
Krok 4.2.1: dopóki (int)stos[g-1]>=48&&(int)stos[g-1]<=57||wyr[i]=='.'
Krok 4.2.1.1: doklej wyr[g-1] do a
Krok 4.2.1.2: g--
Krok 4.2.2: tab[j]=rzutowanie(a)
Krok 4.2.3: j++
Krok 4.2.4: a=""
Krok 4.3: g--

c) Schemat blokowy:



3.7 Obliczanie wartości wyrażenia zapisanego za pomocą odwrotnej notacji polskiej:

a) Specyfikacja wejścia/wyjścia:

Dane: tab[n] - tablica globalna typu double

il - zmienna globalna typu int zliczająca ilość użytych indeksów tablicy tab

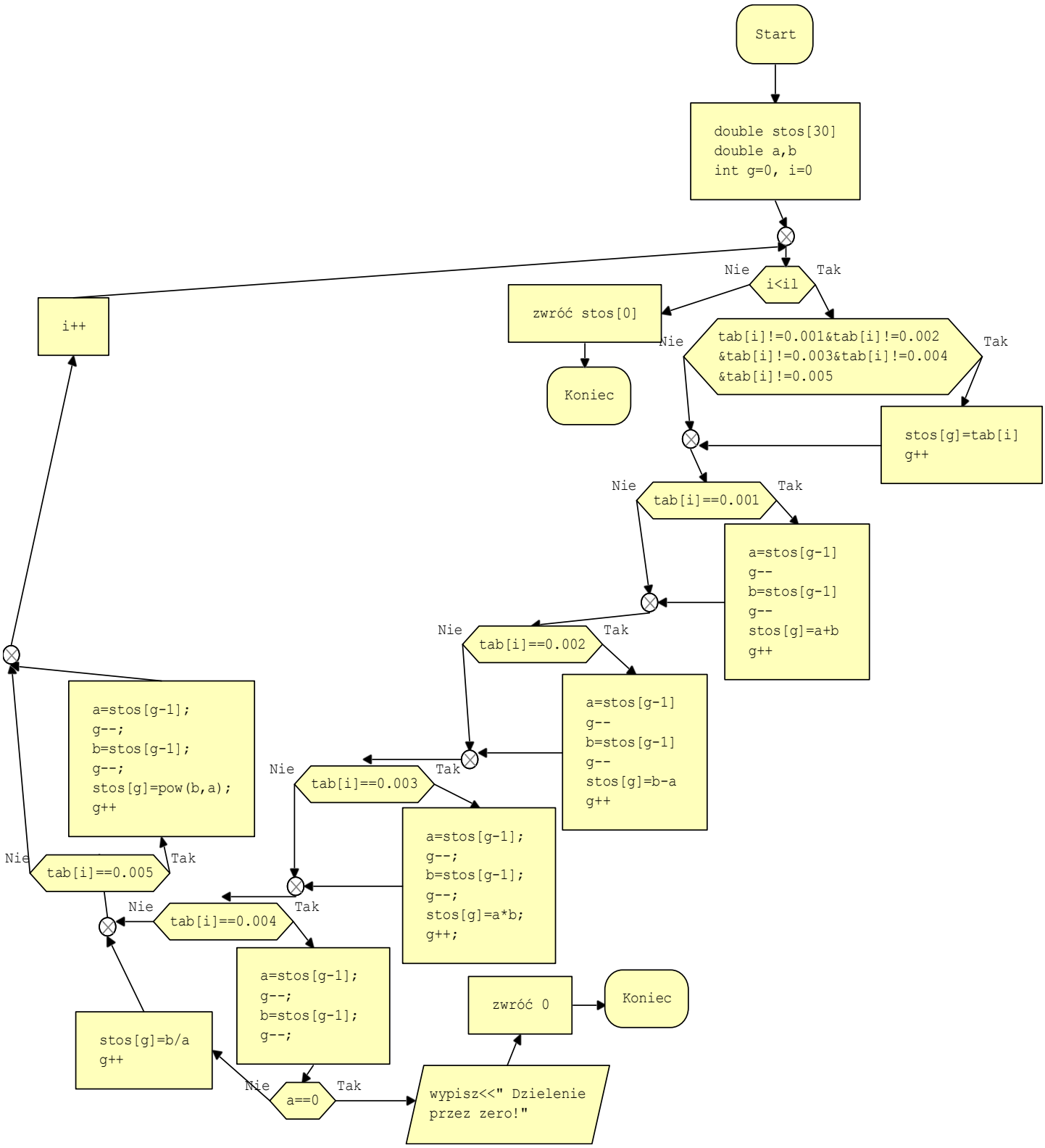
pow - funkcja wykonująca potęgowanie

Wyjście: stos[0] - wartość wyrażenia

b) Pseudokod:

Krok 1: double stos[30],a,b
Krok 2: int g=0,i=0
Krok 3: dopóki i<dl (i=0;i<dl;i++)
Krok 3.1: jeżeli tab[i]!=0.001&& tab[i]!=0.002&& tab[i]!=0.003&& tab[i]!=0.004&& tab[i]!=0.005
Krok 3.1.1: stos[g]=tab[i]
Krok 3.1.2: g++
Krok 3.2: jeżeli tab[i]==0.001
Krok 3.2.1: a=stos[g-1]
Krok 3.2.2: g--
Krok 3.2.3: b=stos[g-1]
Krok 3.2.4: g--
Krok 3.2.5: stos[g]=b+a
Krok 3.2.6: g++
Krok 3.3: jeżeli tab[i]==0.002
Krok 3.3.1: a=stos[g-1]
Krok 3.3.2: g--
Krok 3.3.3: b=stos[g-1]
Krok 3.3.4: g--
Krok 3.3.5: stos[g]=b-a
Krok 3.3.6: g++
Krok 3.4: jeżeli tab[i]==0.003
Krok 3.4.1: a=stos[g-1]
Krok 3.4.2: g--
Krok 3.4.3: b=stos[g-1]
Krok 3.4.4: g--
Krok 3.4.5: stos[g]=b*a
Krok 3.4.6: g++
Krok 3.5: jeżeli tab[i]==0.004
Krok 3.5.1: a=stos[g-1]
Krok 3.5.2: g--
Krok 3.5.3: b=stos[g-1]
Krok 3.5.4: g--
Krok 3.5.5: jeżeli a==0
Krok 3.5.5.1: wypisz „dzielenie przez zero!”
Krok 3.5.5.1: zwróć 0
Krok 3.5.6: stos[g]=b/a
Krok 3.5.7: g++
Krok 3.6: jeżeli tab[i]==0.005
Krok 3.6.1: a=stos[g-1]
Krok 3.6.2: g--
Krok 3.6.3: b=stos[g-1]
Krok 3.6.4: g--
Krok 3.6.5: stos[g]=pow(b,a)
Krok 3.6.6: g++
Krok 4: zwróć stos[0]

c) Schemat blokowy:



4. Przykładowe wyniki działania:

wejście: „-1+2*3.1”
wyjście: 5.2

wejście: „(1+1)/(2-2)”
wyjście: „Dzielenie przez zero!”

wejście: „(10+2”
wyjście: „Błędne nawiasy!”

wejście: „2o^2”
wyjście: „Wyrażenie zawiera niedozwolone znaki!”